UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

18|19

Trabajo fin de grado

Detección sobre vídeo de elementos móviles en zonas restringidas

Rubén Rincón Benito

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Detección sobre vídeo de elementos móviles en zonas restringidas

Autor: Rubén Rincón Benito
Tutor: Pedro Romero Sanz
Ponente: Miguel Ángel García García

junio 2019

Rubén Rincón Benito
Detección sobre vídeo de elementos móviles en zonas restringidas

Rubén Rincón Benito
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos.

El único hombre que no se equivoca
es el que nunca hace nada.

Goethe

Agradecimientos

Me gustaría agradecer a mis amigos que me han acompañado durante estos 5 años y con los que he pasado muy buenos momentos dentro y fuera de la Escuela.

También a mi familia que me han apoyado en todo momento y son una parte muy importante del porqué estoy aquí.

Y por último, pero no menos importante, a todos los profesores tanto de la Escuela Politécnica Superior como de la Facultad de Ciencias por toda la ayuda prestada y la sabiduría transmitida.

Resumen

Estamos en plena revolución industrial, pero esta vez el motor que mueve esta revolución es la tecnología. Empresas e instituciones de todo el globo buscan optimizar sus procesos y para ello se desarrollan cientos de aplicaciones adaptadas a las necesidades de cada una de ellas, teniendo como objetivo principal la máxima integración con su entorno actual.

Este Trabajo de Fin de Grado nace con el objetivo de poder dar solución a algunos de los problemas que distintas entidades puedan estar sufriendo. El proyecto desarrolla una aplicación que, con cualquier cámara sencilla, permita al usuario detectar personas o vehículos en zonas a las que no desea que accedan y recibir un aviso cuando esto suceda.

Para ello, se ha decidido crear una aplicación web para la interfaz, apoyado por un programa desarrollado en Python para la detección de objetos. La aplicación web permite al usuario poder utilizar la herramienta desde prácticamente cualquier dispositivo. Mientras que, por otro lado, el programa de detección separa el trabajo de computación de la aplicación web, facilitando la escalabilidad horizontal.

La aplicación fue pensada con el objetivo de poder detectar a trabajadores en zonas con sustancias o dispositivos que deben ser manejados con extrema seguridad. Por ello, la aplicación permite distinguir a personas dependiendo del color de casco y/o chaleco que lleven puesto, pudiendo estos elementos identificar a los trabajadores en función de su capacitación.

Además, también permite distinguir distintos vehículos, más concretamente, coches, motocicletas, camiones y autobuses. Cada uno de ellos podrá ser identificado por su color, autorizando así solo a los vehículos deseados.

En el presente documento, se muestra todo el proceso de desarrollo de esta herramienta, desde la motivación inicial hasta las pruebas realizadas, pasando por su análisis, diseño, métodos y desarrollo.

Palabras clave

Detección de objetos, visión artificial, OpenCV, web, videovigilancia.

Abstract

We are in the middle of an industrial revolution, but this time the engine that drives this revolution is technology. Companies and institutions from all over the globe are seeking to optimize their processes and, with this in mind, they develop hundreds of applications adapted to the needs of each one of them, having as main objective the maximum integration with its current environment.

This Final Degree Project was born with the aim of being able to solve some of the problems that different entities may be suffering. This project develops an application that, with a simple camera, allows the user to detect people or vehicles in areas that it is forbidden to enter and receive a notification if this happens.

With this purpose, it has been decided to create a web application for the interface, supported by a script developed in Python for the object detection. The web application allows the user to use the tool from virtually any device. While, on the other hand, the detection program separates the computation work from the web application, facilitating horizontal scalability.

The application was designed with the aim of detecting workers in areas with substances or devices that must be handled with extreme safety. Therefore, the application allows to distinguish people depending on the color of helmet and / or vest they wear because these elements can identify workers based on their aptitudes.

In addition, it also allows to distinguish different vehicles, more specifically, cars, motorcycles, trucks and buses. Each of them can be identified by their color, thus authorizing only desired vehicles.

In this document, the entire process of developing this tool is shown, from the initial motivation to the tests carried out, including its analysis, design, methods and development.

Keywords

Object detection, computer vision, OpenCV, web, video surveillance

Índice

Listas

Lista de figuras

1

Introducción

1.1. Motivación

Este proyecto surge debido al trabajo realizado en las prácticas curriculares con proyectos de distintas empresas. En una reunión se buscaron ideas de soluciones que podrían necesitar estos clientes con los que ya se trabajaba en otros proyectos.

El proyecto nace como solución de seguridad para trabajadores que realizan trabajos de manipulación con sustancias peligrosas. El cliente controla estos trabajos mediante la realización de distintos permisos que certifican que los trabajadores están cumpliendo ciertas medidas de seguridad establecidas en él. Estos permisos eran firmados por el dueño de las instalaciones en las que se realiza el trabajo y por los trabajadores a cargo. Ambas partes tenían que cerciorarse de que se estaba cumpliendo el permiso firmado pero la seguridad terminaba en ese punto, no se podía realmente comprobar que se estaban cumpliendo las condiciones establecidas.

Sin embargo, una herramienta así puede tener uso en otras muchas tareas. Entre los cuales se podrían nombrar: playas con bandera roja en las que se quiere vigilar si hay personas que entren en el mar, obras de construcción de edificios u obras en carretera en la cual se quieren detectar vehículos invadiendo el espacio de trabajo. Por lo que se decidió diseñar la aplicación para que fuese adaptable a todos estos casos.

1.2. Objetivos

El objetivo de este trabajo es realizar una aplicación que permita con una cámara poder detectar personas y vehículos con ciertas características que entran en una zona delimitada y recibir un aviso cuando esto ocurra. Tendrá que permitir definir las características, zona y avisos anteriores.

Además, el usuario podrá conectar tantas cámaras como desee, pudiendo configurar cada una de ellas con distintos parámetros. Podrá también ver en directo las cámaras activas que previamente haya configurado. Estas cámaras podrán ser muy sencillas, basta con que puedan conectarse físicamente

a un ordenador o mediante Wi-Fi a una red. En el caso de que estén conectadas físicamente deberán contar con firmware para funcionar como webcam, algo con lo que cuentan prácticamente todas las cámaras actuales. La cámara no se apoyará en otro tipo de sensores como térmicos o radar para la detección de elementos.

Por otro lado, el proyecto comenzaba de cero por lo que el objetivo era poder concluir con una aplicación completamente funcional pero en la que habrá que realizar trabajo futuro (ver apartado 6) para poder obtener los mejores resultados posibles.

1.3. Organización de la memoria

El documento consta de los siguientes 6 capítulos:

- **Capítulo 1:** Introducción
- **Capítulo 2:** Estado del arte
- **Capítulo 3:** Análisis y diseño
- **Capítulo 4:** Desarrollo
- **Capítulo 5:** Integración, pruebas y resultados
- **Capítulo 6:** Conclusiones y trabajo futuro

2

Estado del arte

En este capítulo se van a analizar las alternativas similares a nuestra aplicación que ya existen en el mercado. Se dividirán en función de las características de cada una, ya que no hay ninguna que realice todas las funciones de la aplicación que se presenta en este documento. Estas alternativas se centran en la detección de movimiento, detección de personas o detección de ropa, pero ninguna combina las tres tecnologías.

2.1. Detección de movimiento

Existen un gran número de aplicaciones en el mercado para la detección de movimiento con diferentes características. En esta sección veremos algunas de las aplicaciones que nos ofrecen alguna función interesante además de la propia detección de movimiento.

2.1.1. Eyeline

Eyeline [1] es una aplicación que permite configurar varias cámaras y vigilar todas ellas de forma simultánea. Estas cámaras pueden ser IP por lo que podrán estar conectadas por red sin necesidad de una conexión física con la máquina en la que se ejecuta el software, algo que muchas aplicaciones no permiten.

Además, permite emitir en streaming el vídeo que está siendo tomado en plataformas como Youtube o fijar un horario en el que empezar a grabar. También permite configurar alarmas mediante correo electrónica o SMS.

La aplicación es de pago aunque cuenta con una versión de prueba con funciones limitadas para uso no comercial y solo está disponible para máquinas con sistema operativo Windows.

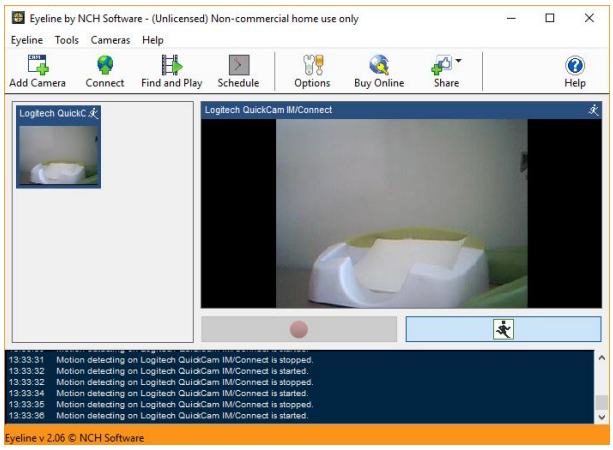

Figura 2.1: Captura de la pantalla principal de Eyeline

2.1.2. TeboCam

Tebocam [2] es una aplicación de detección de movimiento en la que se pueden especificar las zonas en las que queremos que funcione. Estas zonas se podrán delimitar únicamente por un rectángulo que dibujemos sobre el vídeo.

También cuenta con soporte para cámaras IP o alarmas de distintos tipos. Además, Tebocam es completamente gratuito y de código abierto.

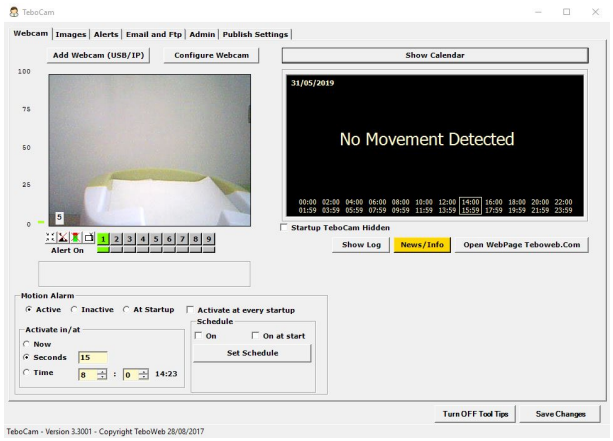

Figura 2.2: Captura de la pantalla principal de Tebocam

2.1.3. UgoLog

UgoLog [3] es otra de las aplicaciones interesantes en el campo de la detección de movimiento. En este caso, la característica que la hace diferente es que no es necesario instalar ninguna aplicación en el ordenador.

UgoLog es una aplicación web, una vez registrado, se pueden configurar y visualizar las cámaras

directamente desde el navegador. Esto permite una gran movilidad ya que puedes acceder desde cualquier dispositivo con acceso a la red y un navegador.

Por otro lado, es más simple que las aplicaciones anteriores y las opciones de configuración son más reducidas. Además, la versión gratuita es muy limitada tanto como en funcionalidad como en tiempo de uso. También dispone de versión de escritorio para Windows.

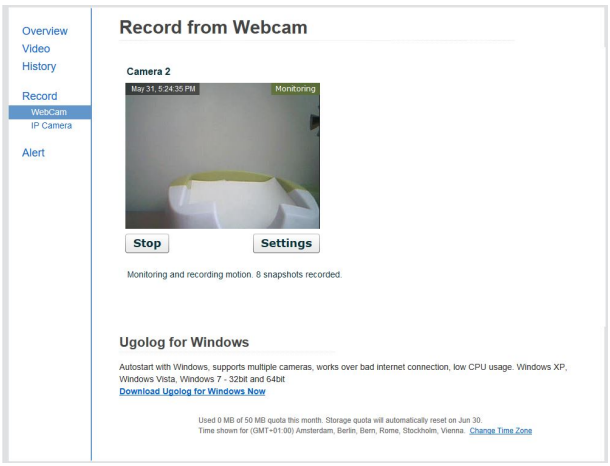

Figura 2.3: Captura de la pantalla de grabación de UgoLog

2.2. Detección de personas

En esta sección analizaremos algunas de las soluciones que cuentan con detección de personas sobre la grabación en directo. Estas aplicaciones van un paso más allá que las vistas en la última sección, por ello, los requisitos tanto hardware como económicos aumentan en consecuencia.

2.2.1. Sighthound

Sighthound [4] es una de las empresas importantes en el sector de la visión artificial. En este caso, vamos a analizar su aplicación de vigilancia Sighthound Video [5].

Ellos dicen en su página web que son la alternativa cuando OpenCV es demasiado lento para las necesidades de sus clientes. Apoyados en redes neuronales propietarias dicen que consiguen mejorar el rendimiento de la competencia con menos datos de entrenamiento [6].

Su aplicación nos ofrece la posibilidad de configurar zonas y detectar cuando hay personas u objetos que entran o salen de la zona. Además podemos vigilar todas las cámaras activas en un mismo cuadro o buscar las incidencias que han ocurrido en cada cámara configurada y ver la captura asociada al suceso.

Las pruebas realizadas muestran que no reconoce en muchas ocasiones una persona cuando solo una parte de ella (cabeza, brazos...) aparece en la grabación.

Es una aplicación de pago con versión gratuita limitada y cuenta únicamente con versiones para Windows y MacOS.

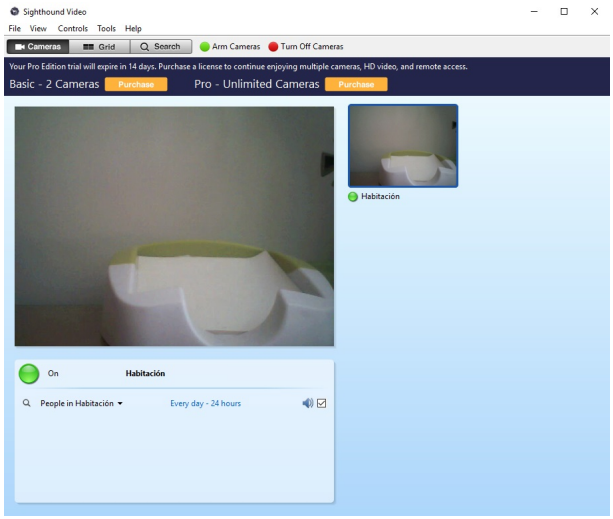

Figura 2.4: Captura de la pantalla de cámaras de Sighthound

2.2.2. Presence

Presence [7] no es una aplicación, en este caso, Presence es una cámara de seguridad de la empresa Netatmo [8] que detecta personas, animales y vehículos. Es una cámara IP y es manejable desde su aplicación para multitud de dispositivos.

Desde estas aplicaciones se podrán configurar zonas, alarmas u objetos a detectar. Además, la cámara cuenta con sensores infrarrojos y un foco para ayudar a la detección nocturna. Su precio es de 299,99 euros [9].

Figura 2.5: Cámara Presence de Netatmo

2.3. Detección de ropa

Por último, analizaremos el estado del arte de la detección de vestimenta. Este es un área menos desarrollada comercialmente por lo que mostraremos un único ejemplo de investigación relacionado con el objetivo de este trabajo. También veremos en el ejemplo el uso de la detección de color, la cual tiene una gran importancia a la hora de detectar elementos como las prendas que lleva una persona.

Vamos a resumir brevemente las soluciones propuestas en [10] para detectar en tiempo real, sobre cámaras de vigilancia, la ropa que visten las personas que aparecen en la imagen.

Para realizar esta tarea, se divide el trabajo en dos partes, la segmentación y detección de cada prenda que lleva una persona, y posteriormente, el análisis de este resultado para obtener qué tipo de prendas se han reconocido.

La segmentación es realizada mediante métodos de segmentación por color, es decir, aplicar los cambios de color entre prendas para poder establecer un primer borde de separación entre ellas. Posteriormente, se aplican técnicas como el diagrama de Voronoi o Canny edge para terminar de definir los bordes con la mayor precisión posible.

Por otro lado, para el reconocimiento de las prendas se usan algoritmos de aprendizaje supervisado como SVM, para el cual se escogen prendas de las cuales se ha obtenido un resultado de alta calidad en la segmentación inicial para usarlas como entrenamiento. Datos como la cantidad de piel que queda sin cubrir por la prenda son muy útiles para clasificarlas en los distintos tipos.

3

Análisis y diseño

En este capítulo describiremos la fase de análisis y diseño de la aplicación, desde la idea inicial hasta el último detalle de la aplicación en su diseño final.

En primer lugar, se enumerarán los requisitos que se fijan para el proyecto y la arquitectura general que estructurará la aplicación. Posteriormente, se definirán las tecnologías que se usarán durante el desarrollo y el motivo de su elección.

3.1. Análisis de requisitos

A continuación pasamos a detallar los requisitos que se establecieron para la aplicación. Estos requisitos se fijaron a lo largo de varias reuniones en las que se pensó en los problemas que los clientes no podían resolver con aplicaciones actuales, además de pensar en posibles problemas que se podían resolver con una aplicación así, tales como la vigilancia de obras en carreteras o la vigilancia de playas durante un temporal.

Los requisitos se dividirán en requisitos funcionales, en los que se detallarán las funcionalidades que debe de tener la aplicación y requisitos no funcionales, en los que se describen los estándares de calidad a los que se debe someter la aplicación.

3.1.1. Requisitos funcionales

[RF 01] Cualquier usuario en posesión de una cuenta podrá iniciar sesión en el sistema.

Requisitos de configuración de cámaras

[RF 02] Un usuario podrá añadir tantas cámaras IP como quiera a la base de datos de la aplicación.

[RF 03] El usuario podrá definir mediante tantos puntos como sea necesario una zona poligonal en cuyo exterior quedará anulada la vigilancia.

[RF 04] El usuario podrá elegir si desea detectar personas, vehículos o ambos.

[RF 05] En caso de que el usuario desee detectar personas, podrá configurar si desea detectar:

1.– Cualquier persona

2.– Persona con casco de un determinado color

3.– Persona con chaleco de un determinado color

4.– Persona que cumpla los puntos anteriores 2 y 3

[RF 06] En caso de que el usuario desee detectar vehículos, podrá configurar si desea detectar:

- Coches
- Motocicletas
- Camiones

[RF 07] El usuario podrá elegir un color para los vehículos a detectar [RF 06] si así lo desea.

[RF 08] El usuario podrá establecer alarmas cuando se detecte un objeto no deseado. Estas alarmas podrán ser mediante:

- Correo electrónico
- Alerta web
- Notificación del sistema (API notificaciones HTML5)

[RF 09] El usuario podrá elegir si desea activar la alarma [RF 08] cuando parte del objeto esté en la zona [RF 03] o solo cuando el objeto al completo esté dentro de ella.

[RF 10] Un usuario podrá configurar en una cámara previamente añadida [RF 02] las opciones de vigilancia [RF 03-09] que desee y comenzar la vigilancia.

Requisitos de visualización de cámaras

[RF 11] El usuario podrá terminar la vigilancia sobre cualquier cámara activa [RF10]

[RF 12] El usuario podrá visualizar cualquier cámara activa.

[RF 13] En la pantalla de visualización de la cámara se deberán mostrar las opciones elegidas en su configuración. Estos detalles incluirán:

- Modo de funcionamiento [RF 09]
- Clases a detectar [RF 04]
- Objetos a detectar y sus colores [RF 05-07]
- Alarmas configuradas [RF 08]

[RF 14] El usuario podrá pausar en cualquier momento el vídeo en directo.

3.1.2. Requisitos no funcionales

[RNF 01] La aplicación deberá ser accesible tanto desde las versiones de escritorio como móviles de los principales navegadores.

[RNF 02] La aplicación contará con una pequeña ayuda para conocer el funcionamiento de esta.

[RNF 03] La aplicación debe de ser capaz de hacer funcionar al menos una cámara de vigilancia en máquinas con tarjetas gráficas de gama baja de uso doméstico.

[RNF 04] La aplicación deberá mostrar mensajes de error descriptivos en caso de que falle en su ejecución.

3.2. Arquitectura

En la siguiente figura se muestra cómo se estructurará la aplicación en diferentes módulos. Este esquema se realiza acorde al análisis previo para cumplir así todos los requisitos impuestos. Posteriormente, se analizarán las distintas opciones en cuestión de tecnologías para poder desarrollar toda esta funcionalidad.

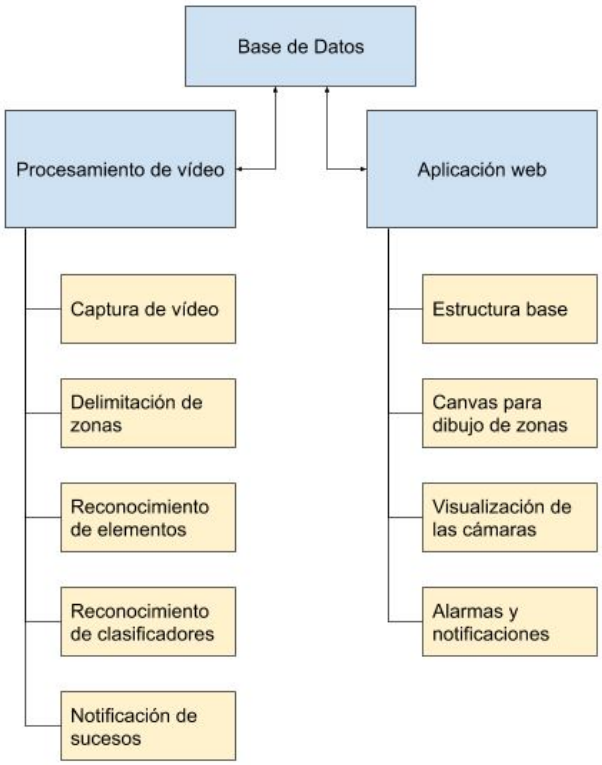

Figura 3.1: Esquema de la arquitectura de la aplicación

Como se puede ver en la figura la aplicación se compondrá de tres módulos principales:

- **Aplicación de procesamiento de vídeo:** Se encarga de todo lo relacionado con el vídeo y la detección de los elementos en él.
- **Aplicación web:** Se encarga de proporcionar al usuario las herramientas necesarias para la configuración de todos los parámetros necesarios para el funcionamiento correcto de la aplicación.
- **Base de datos:** Se encarga de proporcionar un canal de comunicación entre ambas aplicaciones. También almacenará datos permanentes como los datos de usuario o las direcciones IP de las cámaras añadidas por cada uno de los usuarios.

De estos tres módulos principales dependen 9 submódulos que se encargarán de cada una de las tareas o funcionalidades a implementar. Las tareas dependientes del módulo de procesamiento de vídeo son las siguientes:

- **Captura de vídeo:** Este módulo se encarga de conectar el dispositivo de grabación de vídeo externo a la aplicación de procesamiento de vídeo.
- **Delimitación de zonas:** Este módulo se encarga de recibir las coordenadas de la zona establecidas mediante la aplicación web y generar la zona restringida correspondiente para su posterior uso.
- **Reconocimiento de elementos:** Este módulo se encarga de la detección de elementos en el vídeo. Los elementos a detectar serán indicados en la aplicación web donde se podrá elegir entre personas o coches.
- **Reconocimiento de clasificadores:** Este módulo se encarga de identificar sobre los elementos anteriormente detectados la existencia de clasificadores que serán utilizados para diferenciar elementos en su acceso a las zonas delimitadas. Estos clasificadores podrán ser cascos y chalecos de distintos colores en personas y el color del vehículo para coches.
- **Notificación de sucesos:** Este módulo se encarga de detectar cuando un elemento ha ingresado a una zona que a la que no tiene permiso para acceder. Este acceso no permitido será notificado a los distintos dispositivos que se hayan configurado desde la aplicación web. Estas notificaciones podrán ser alarmas sonoras, correos electrónicos o notificaciones en la aplicación web.

Por otro lado, las tareas dependientes de la aplicación web son:

- **Estructura base:** Este módulo contendrá la estructura principal de la aplicación web, se encargará de dar soporte a todas las funcionalidades adicionales que se ejecutarán sobre esta.
- **Canvas para dibujo de zonas:** Este módulo se encarga de recibir y codificar las zonas delimitadas por el usuario mediante el dibujo de ellas sobre el vídeo. Estas zonas pueden tener diferentes formas y estas formas deben ser traducidas a un lenguaje común que la aplicación de procesamiento de vídeo pueda entender para la definición de la zona restringida final.
- **Visualización de las cámaras:** Este módulo se encarga de mostrar la grabación en directo de las cámaras al usuario. Sobre la grabación se mostrará la zona definida por el usuario. Además, también se mostrará los detalles de la configuración de la detección realizada y las opciones para pausar o finalizar la grabación.
- **Alarmas y notificaciones:** Este módulo se encarga de definir las diferentes alarmas y notificaciones que el usuario desea para que la aplicación de procesamiento de vídeo pueda enviar estas cuando se detecte un suceso no deseado. También se encarga de recibir y mostrar al usuario las notificaciones web si este así lo ha especificado en la definición inicial.

3.3. Análisis de las soluciones

En esta sección discutiremos las soluciones propuestas para llevar a cabo el desarrollo de la aplicación, las que finalmente se escogieron y el motivo de la decisión. Seguiremos el esquema de la arquitectura de la sección anterior para definir la solución tecnológica precisa a cada uno de los problemas planteados.

3.3.1. Aplicación web

La aplicación web servirá como interfaz para el usuario. Este podrá configurar desde un navegador todas sus cámaras conectadas y ver la grabación en directo de cualquiera de ellas. A continuación mostramos las tecnologías usadas en el desarrollo y el porqué de su elección.

Estructura base

La aplicación web ha sido realizada con el framework Django. Esto nos permite trabajar de una forma sencilla y eficaz en el desarrollo de todas las características de la aplicación al compartir lenguaje de programación con la aplicación de procesamiento de vídeo.

Además, se han usado HTML, Javascript y CSS para el desarrollo de las páginas web. También se ha usado la librería Bootstrap para trabajar con ellos y conseguir un resultado adecuado en el diseño.

Canvas para el dibujo de zonas

El canvas será el lugar donde el usuario podrá ver una captura de la visión de la cámara y definir mediante una serie de puntos un polígono una zona en la que se desea restringir el acceso a las personas o vehículos.

Esto se conseguirá mediante el uso del elemento <canvas> de HTML5. Junto con el uso de Javascript podremos dibujar en el canvas, sobre la captura de la cámara, unos puntos unidos mediante aristas que al cerrarse mostrarán el polígono definido por el usuario. El área de este polígono quedará sombreado con color para que el usuario pueda visualizar la zona definida y borrarla para empezar de nuevo si no ha conseguido el resultado deseado.

Alarmas y notificaciones

Las notificaciones que deberán manejar la aplicación web serán el envío de alertas mediante la función alert() de Javascript. Además, también deberá enviar las notificaciones HTML5 o notificaciones “push”, esto se realizará mediante la API [11] oficial para ello.

3.3.2. Procesamiento de vídeo

Por otro lado, hablaremos sobre las decisiones tomadas respecto al procesamiento de vídeo. La idea inicial era usar procesamiento paralelo mediante hilos para dar soporte a cada una de las cámaras. Estos hilos tendrían como padre al propio proceso de Django.

Sin embargo, esta opción se descartó, ya que no se puede asegurar que las peticiones sean atendidas siempre por el mismo proceso. Además, un fallo en la página web supondría la pérdida de datos en el procesamiento de vídeo y determinadas mejoras o funcionalidades futuras se verían frenadas por la unión inseparable entre aplicación web y procesamiento de vídeo.

Por ello, se optó por separar en procesos cada una de las cámaras de vigilancia activas. Esto además permite una mayor escalabilidad horizontal en el futuro, pudiendo así separar en distintos servidores los procesos de vídeo cuando sea necesario por un límite en la capacidad de procesamiento.

Este proceso asociado a cada una de las cámaras será un script de Python que se encargará de realizar todo el trabajo de procesamiento ayudado por diversas librerías que iremos mencionando en los siguientes apartados. Este script recibirá mediante argumentos de ejecución la información necesaria de la configuración realizada por el usuario en la web.

Captura de vídeo

La captura de vídeo se realizará mediante cámaras IP ya que elimina la necesidad de una conexión física con el servidor. Aun así, también se han realizado pruebas con cámaras mediante conexión USB y podrán ser utilizadas en futuras mejoras.

El uso de la cámara por la aplicación se realiza mediante la librería OpenCV, en particular, con la clase VideoCapture() que permite el uso tanto de webcams integradas, cámaras USB o cámaras IP.

Delimitación de zonas

El script de detección recibirá mediante argumentos las coordenadas de la zona que ha marcado el usuario sobre el canvas de dibujo en la web. Estas coordenadas son parseadas y con ellas se realiza un contorno sobre la imagen con la librería OpenCV simulando la zona dibujada por el usuario que permitirá más tarde analizar los cambios en la imagen que ocurran dentro de este contorno.

Reconocimiento de elementos

El reconocimiento de personas y vehículos se realiza mediante el uso de la librería ImageAI [12] que hace uso de otras librerías como TensorFlow o Keras para proporcionar un acceso sencillo a la detección de objetos.

En el futuro, es posible que se haga la transición a un uso directo de estas librerías, ya que permiten una mayor libertad y además, son librerías ampliamente conocidas y empleadas en todo tipo de proyectos, contando así con un mayor soporte e información disponible.

Reconocimiento de clasificadores

En el reconocimiento de clasificadores, es decir, del color de los cascos, chalecos y vehículos se usan las librerías OpenCV y Numpy. La librería OpenCV permite realizar la detección por color, pudiendo usar distintos espacios de color como RGB, HSV o LAB. Por otro lado, Numpy nos permite manipular la imagen mediante sus pixeles, pudiendo así tomar zonas determinadas que nos interesan para su análisis.

Notificación de sucesos

Cuando un objeto no deseado se encuentre en la zona restringida el script escribirá en base de datos para indicarlo y enviará las alertas que estén a su alcance, por ejemplo, los correos electrónicos. En caso contrario, la aplicación web leerá el cambio en base de datos y notificará las alarmas restantes mediante las vías elegidas por el usuario.

3.3.3. Base de datos

Para la base de datos, la idea inicial era utilizar Redis o Memcached, ya que se adaptaban perfectamente a las necesidades de la aplicación. Sin embargo, la solución elegida fue MySQL dado que el número de operaciones no iba a ser excesivamente alto como para dar el salto a una de las alternativas iniciales y la velocidad no iba a ser un problema. Además, se contaba con mayor experiencia en su uso.

4

Desarrollo

En este capítulo se explicarán paso a paso todas las funcionalidades de la aplicación. Para ello, empezaremos como cualquier usuario que utilice la aplicación, por la web. A partir de la aplicación web, profundizaremos en el procesamiento de vídeo y cómo es este realizado. Por último, se describirá la base de datos y su modelo, parte fundamental para la comunicación dentro de la aplicación.

4.1. Aplicación web

Lo primero que se encuentra el usuario al entrar en la web en la pantalla de login. En esta pantalla se pide el usuario y contraseña que deberá haber sido obtenido previamente contactando a los administradores de la aplicación. Por el momento, no existe la opción de registro de un usuario.

Para este login se ha utilizado el sistema de autenticación de Django [13] que permite la creación y manejo de usuarios desde el panel de administración de Django y la integración con la base de datos elegida para el proyecto.

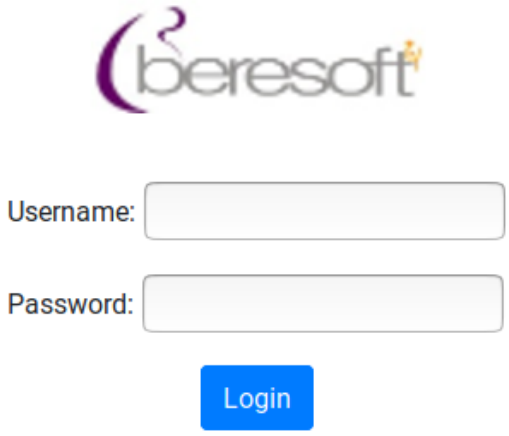

Figura 4.1: Captura de la pantalla de login de la aplicación web

Una vez se haya iniciado sesión el usuario se encontrará con la página principal con un menú superior con las distintas opciones. En esta pantalla principal se muestra una imagen con instrucciones

básicas sobre cómo utilizar la aplicación y navegar por sus pantallas.

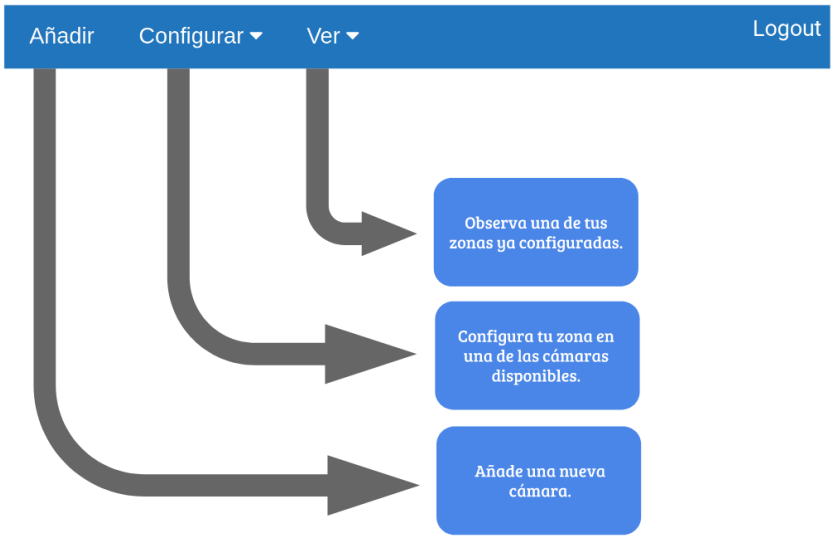

Figura 4.2: Instrucciones mostradas en página principal

En el menú superior nos encontramos con tres opciones: Añadir, Configurar y Ver. Primero añadiremos una cámara, que después configuraremos y por último podremos visualizar.

4.1.1. Añadir una cámara

Seleccionando la opción “Añadir” se muestra una ventana flotante para introducir la información de nuestra cámara IP y el nombre que queremos asignar a esta cámara para poder reconocerla posteriormente. Esta información queda guardada en la base de datos para el uso en otras sesiones.

Figura 4.3: Ventana flotante desplegada para añadir una cámara

Una vez añadida la cámara, podremos pasar a configurar la detección que queremos realizar.

4.1.2. Configurar una cámara

Para configurar una cámara bastará con poner el cursor sobre la opción Configurar del menú superior y se desplegará una lista con las cámaras disponibles para configurar, es decir, las añadidas previamente y que no están activas.

Una vez seleccionada la cámara deseada se nos mostrará la pantalla de configuración [Figura 4.4]. Desde esta pantalla podremos ver una captura de la visión de la cámara sobre la que podremos dibujar una zona seleccionando “Dibujar línea”, entonces podremos seleccionar tantos puntos como sean necesarios sobre la imagen que serán los vértices del polígono que definirá la zona.

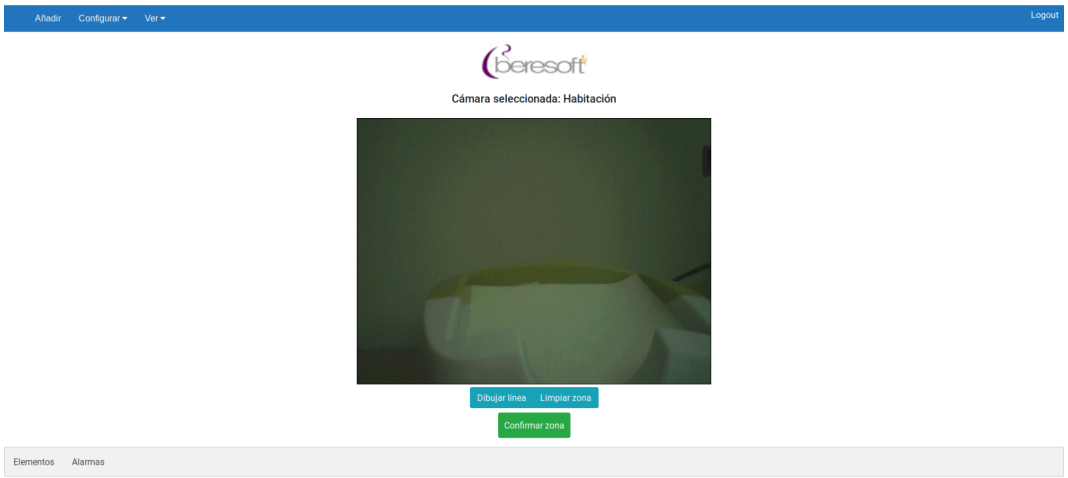

Figura 4.4: Captura de la pantalla de configuración de la aplicación web

Estos puntos se dibujan en la imagen usando Javascript sobre el canvas HTML, una vez que el polígono se cierra se puede ver la zona sombreada [Figura 4.5] en la que se restringirá el acceso a los objetos que seleccionemos a continuación.

Cuando hemos definido la zona deseada, podremos configurar los elementos a detectar. Abriendo la pestaña “Elementos” en la parte inferior se nos muestra un selector donde podemos seleccionar si queremos detectar personas, vehículos o ambas. También debemos seleccionar el modo de funcionamiento, completo, si queremos que el objeto completo deba estar en la zona para activar la alarma o parcial, si queremos que únicamente baste con una parte del objeto dentro de la zona para activarla.

Una vez seleccionado la detección de personas o vehículos se desplegarán las opciones correspondientes en función de la elección [Figura 4.6]. En estas opciones podemos elegir los elementos a detectar y los colores de estos elementos. Estos colores también se pueden seleccionar mediante un cuentagotas sobre la captura en la que dibujamos la zona, esto elimina por completo el problema del cambio de color que aplica el sensor de la cámara sobre el color que percibe el ojo humano. En caso contrario, este problema se arreglará en la medida de lo posible en el procesamiento de vídeo, que explicaremos en la siguiente sección.

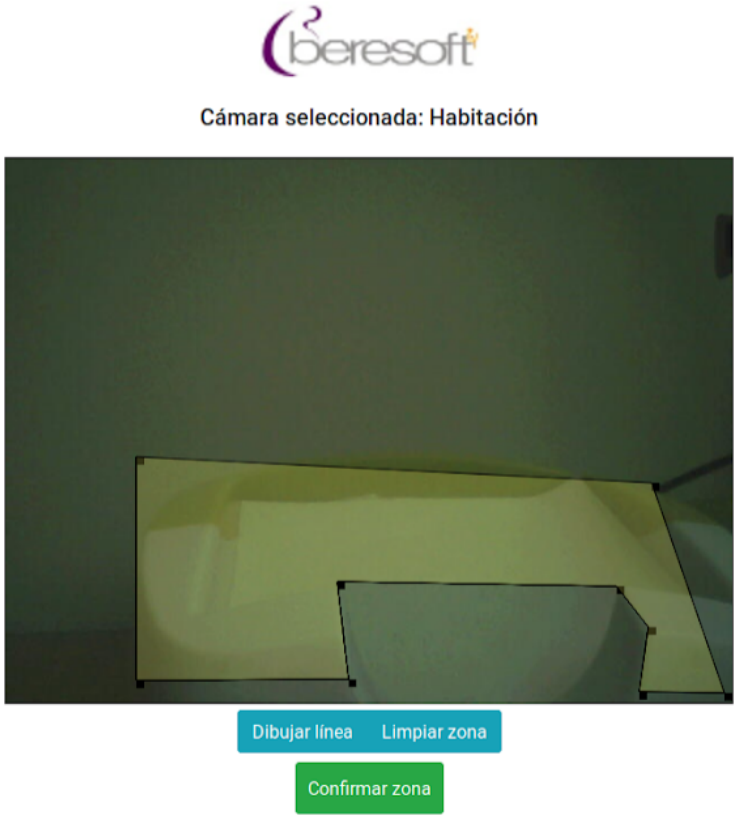

Figura 4.5: Captura del canvas después de que el usuario defina la zona

Figura 4.6: Opciones de personalización de la detección de personas y vehículos

Por último, seleccionaremos en la pestaña “Alarmas” las notificaciones que desea recibir el usuario. Además, el usuario podrá elegir el número de segundos que necesita estar el objeto dentro de la zona para activar la alarma y los segundos a esperar entre la desactivación de una alarma y la activación de la siguiente, para evitar varias notificaciones en un periodo corto de tiempo si el objeto entra y sale de la zona continuamente.

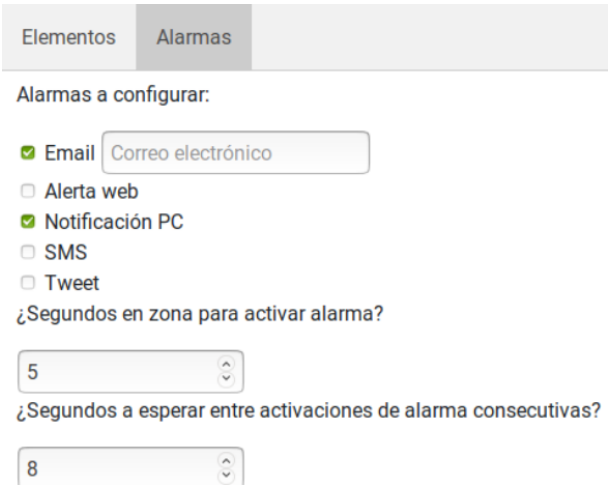

Figura 4.7: Opciones de personalización de las alarmas y notificaciones

Ahora, podemos seleccionar el botón “Confirmar zona” que guardará la configuración realizada y ejecutará un script de Python que será el encargado de la detección y procesamiento del vídeo recogido. Este proceso será completamente independiente de la aplicación web y quedará registro por su identificador en la base de datos. Esto permite asociar a cada usuario una serie de procesos de vigilancia activos los cuales podrá visualizar [Figura 4.8].

Toda la configuración realizada será enviada al nuevo proceso mediante argumentos de ejecución, entre ellos están las coordenadas, los elementos a detectar o los colores elegidos.

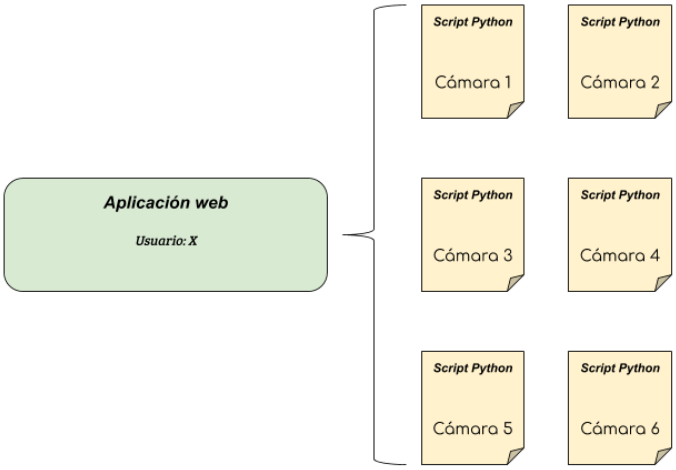

Figura 4.8: Esquema de relación entre aplicación web y aplicación de procesamiento de vídeo

4.1.3. Visualizar una cámara

Una vez hemos configurado una cámara podremos acceder a su visualización desde el menú superior [Figura 4.9] en el que se nos mostrarán todas las cámaras activas en ese momento.

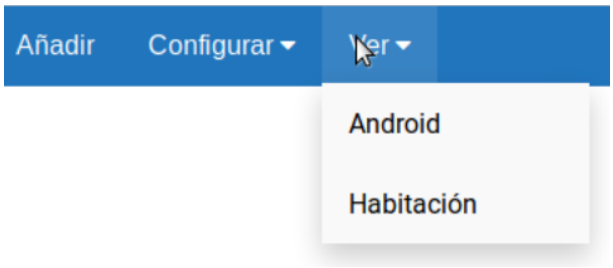

Figura 4.9: Acceso mediante el menú superior a cualquier cámara activa

En esta pantalla [Figura 4.10] podremos ver el vídeo en directo, además se mostrará sobrepuesta la zona definida en la configuración. El resto de los parámetros de la configuración los podemos ver en el recuadro de la parte derecha. En la parte inferior, disponemos del botón rojo para finalizar la grabación, acompañado de los botones de play y pausa.

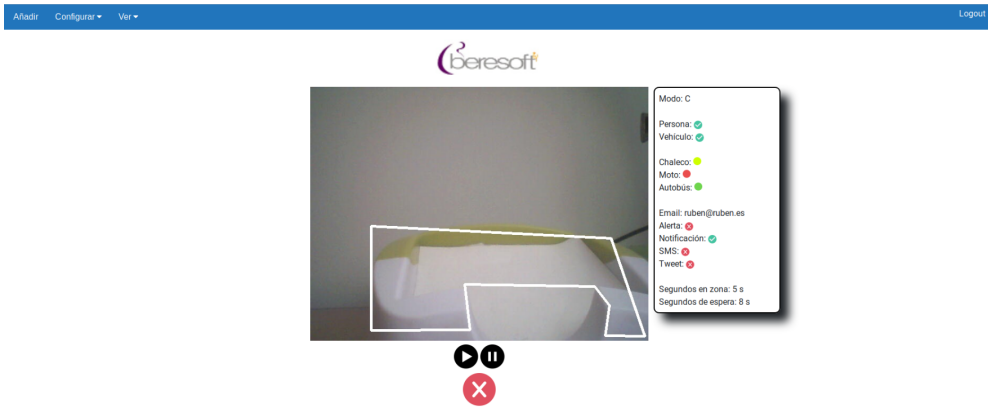

Figura 4.10: Pantalla de visualización de la cámara en vivo

4.2. Procesamiento de vídeo

Como hemos visto en la sección anterior, al seleccionar la opción de “Confirmar zona” se lanza un proceso independiente para el análisis de vídeo. Este proceso es un script de Python que se encarga de leer imágenes de la cámara, detectar objetos sobre esta imagen, aplicar la zona restringida y analizar el color de los identificadores como cascos, chalecos o vehículos.

4.2.1. Argumentos y lectura de cámara

El script recibe argumentos de ejecución con todos los parámetros configurados por el usuario en la aplicación web. Estos argumentos son leídos y asignados con ayuda de la librería *argparse* [14].

Una vez asignados los argumentos, el programa empieza a leer imágenes de la cámara. El modo de realizar esta lectura es tomar una captura al principio de cada bucle de procesamiento, es decir, se toma una captura y se realiza la detección de objetos en la zona restringida y la detección de color, una vez tenemos un resultado, se vuelve al principio del bucle.

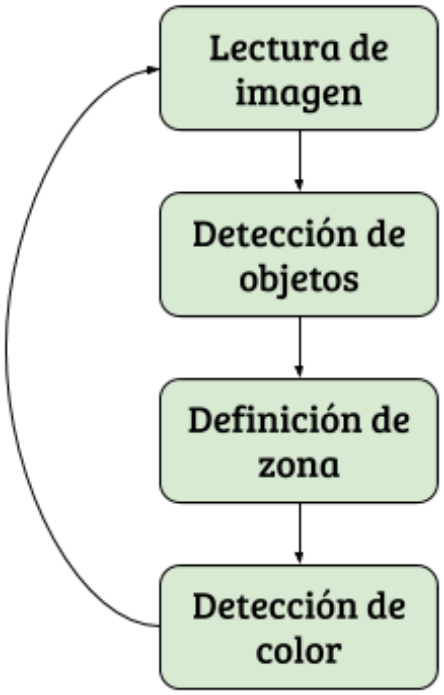

Figura 4.11: Bucle de funcionamiento de la aplicación de procesamiento de vídeo

4.2.2. Detección de objetos

Sobre la imagen capturada detectamos los objetos que aparecen en ella, para ello primero tenemos que fijar los elementos escogidos por el usuario para ser detectados. Después la librería ImageAI nos devuelve las coordenadas de un rectángulo que rodea a cada uno de los objetos detectados.

En caso de que no detecte ningún objeto se pasa a la siguiente iteración del bucle. En caso contrario, se realiza un contorno [15] con la librería OpenCV que más tarde usaremos para calcular su posición dentro de la imagen.

Figura 4.12: Detección de un coche en un frame de un vídeo

4.2.3. Zona restringida e intersección

La zona restringida viene definida mediante el dibujo que realiza el usuario en el canvas de la aplicación web, las coordenadas de los puntos de este dibujo son enviadas al programa de procesamiento mediante argumento. Con estas coordenadas realizamos un contorno de OpenCV sobre la imagen que nos servirá para realizar una intersección [Figura 4.13] con cada uno de los objetos detectados previamente.

Esta intersección se hará mediante la función *logical_and* de Numpy en la que se comprobará si el número de píxeles que comparten ambos contornos es distinto de cero. En esta parte también se tendrá en cuenta el modo de funcionamiento elegido por el usuario, ya que si se ha elegido el modo completo, el tamaño de la intersección entre la zona y el objeto tendrá que ser igual al tamaño del objeto.

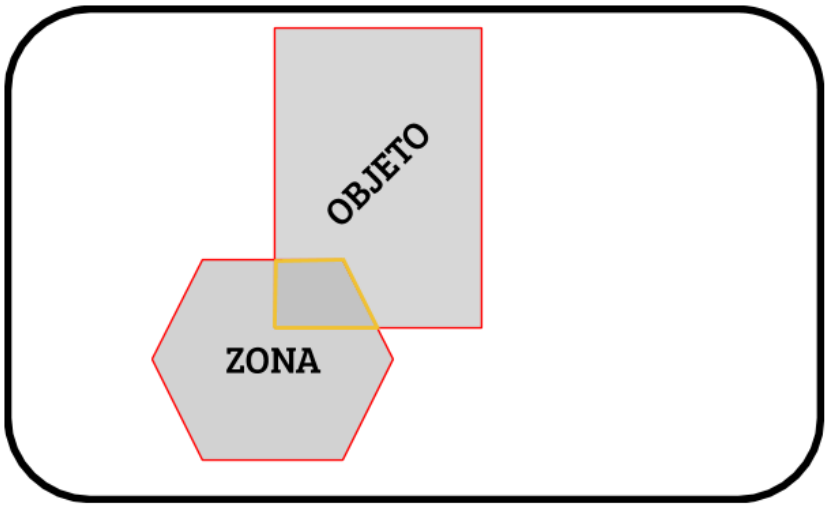

Figura 4.13: Contorno en amarillo que será resultado de la intersección de zona y objeto

En caso de que la intersección de la zona y el objeto cumpla las condiciones de tamaño necesarias se comprobará que los tiempos marcados por el usuario para la activación de la alarma se cumplan. Estos tiempos se controlan mediante temporizadores ejecutados en hilos en paralelo.

En el caso del temporizador de los segundos en zona necesarios para la activación, éste se activa en la primera detección del objeto y se reinicia en caso de que el objeto deje de estar en la zona. Por otra parte, en el caso del temporizador de espera entre alarmas, el temporizador se activa cuando la alarma se desactiva y no se reinicia en ningún caso. Por lo que, si un objeto es detectado dentro de la zona y ambos temporizadores han finalizado se envían las imágenes detectadas al procesamiento de color, que será el último paso antes de activar la alarma.

4.2.4. Espacios de color e identificadores

Una vez que se ha comprobado que existe un objeto dentro de la zona se debe detectar si los identificadores de color de este objeto cumplen con las reglas impuestas por el usuario. Para ello, se analizará el color seleccionado por el usuario dentro de la imagen aplicando un rango para así evitar ignorar colores que sí son los seleccionados por el usuario o muy similares a ellos.

Esto es así, ya que el sensor de la cámara y las luces y sombras de la imagen pueden hacer cambiar mucho un color respecto a cómo es recibido por el ojo humano. El problema es que el espacio de color por defecto RGB no es un buen sistema para realizar estos rangos ya que añadir o restar valor a los colores puede darte como resultado colores muy distintos.

Por ello, tras realizar pruebas con distintos espacios de color [Figura 4.14] se decidió usar el espacio LAB, en concreto, CIELAB. Este espacio de color permite manejar por separado la luminosidad (canal L) del color (canales A y B), con lo que conseguimos corregir los defectos o diferencias entre el ojo humano y el sensor de la cámara de una forma más precisa. Además, al ser la luminosidad un canal separado, podemos hacer el rango más amplio en este canal que en los canales de color, permitiendo así detectar con más exactitud los colores que queremos tanto en condiciones de baja como de alta luminosidad. Esta diferencia de rangos no tenía sentido realizar en el espacio RGB, ya que los tres canales representaban color.

Una vez hemos transformado nuestra imagen a LAB se comprueba para cada uno de los identificadores, casco y chaleco en caso de personas y carrocería en caso de vehículos, si el porcentaje de color en determinadas zonas es mayor que un cierto mínimo requerido. Por ejemplo, para la detección de color en cascos se busca el color elegido en el primer 10 % del alto de la imagen [Figura 4.15], pero este porcentaje cambia en función de las dimensiones del rectángulo que rodea a la persona para adaptarse a la posición donde se encuentra el casco en la imagen.

Si se ha encontrado el color elegido en los identificadores se activa la alarma en la base de datos, de donde leerá este cambio la aplicación web para notificar al usuario. Veremos cómo funciona esta

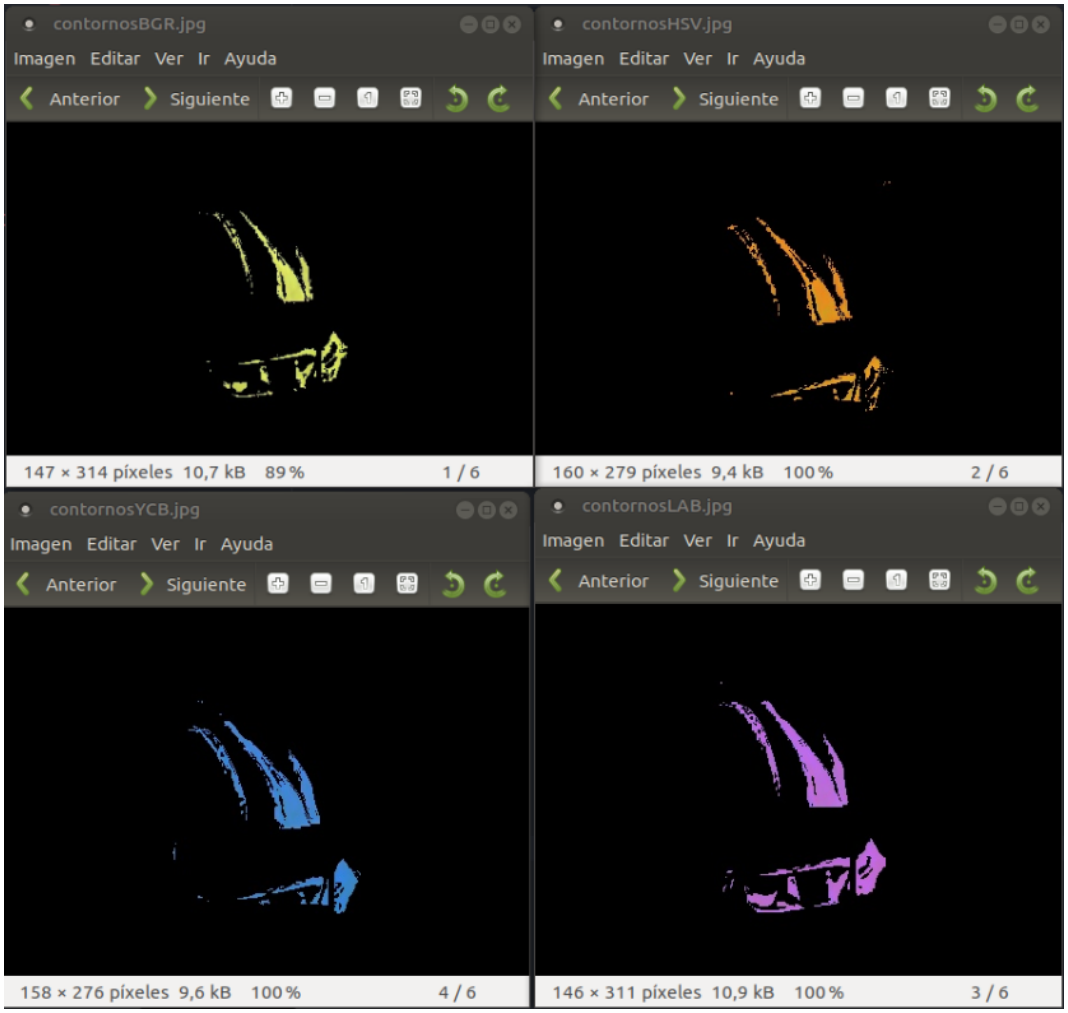

Figura 4.14: Comparación del área de chaleco detectada con diferentes espacios de color. RGB=Amarillo, HSV=Naranja, YCB=Azul, LAB=Morado.

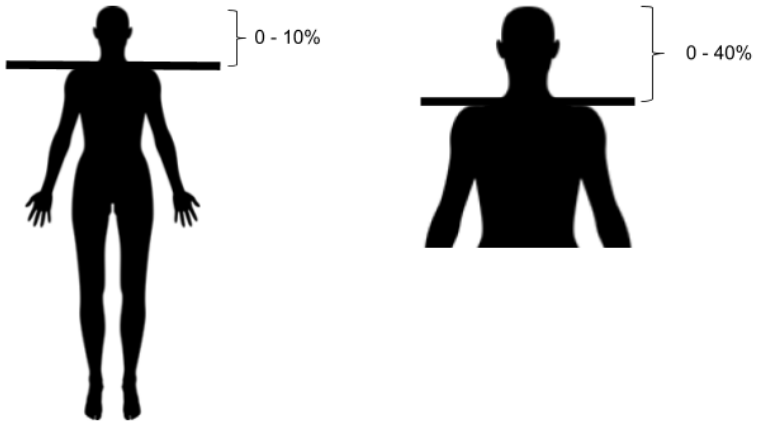

Figura 4.15: Área de la imagen en la que se busca el color del casco

activación más en detalle en la siguiente sección.

4.3. Base de datos

La base de datos es el elemento que conecta ambas aplicaciones, la aplicación web y la aplicación de procesamiento de vídeo. Mediante ella se transmite la información que necesita saber cada aplicación sobre la otra, además de guardar todos los datos relevantes que el usuario introduce en la aplicación como la dirección IP de sus cámaras. En la siguiente figura se muestra un diagrama con las diferentes tablas que componen la base de datos y sus relaciones.

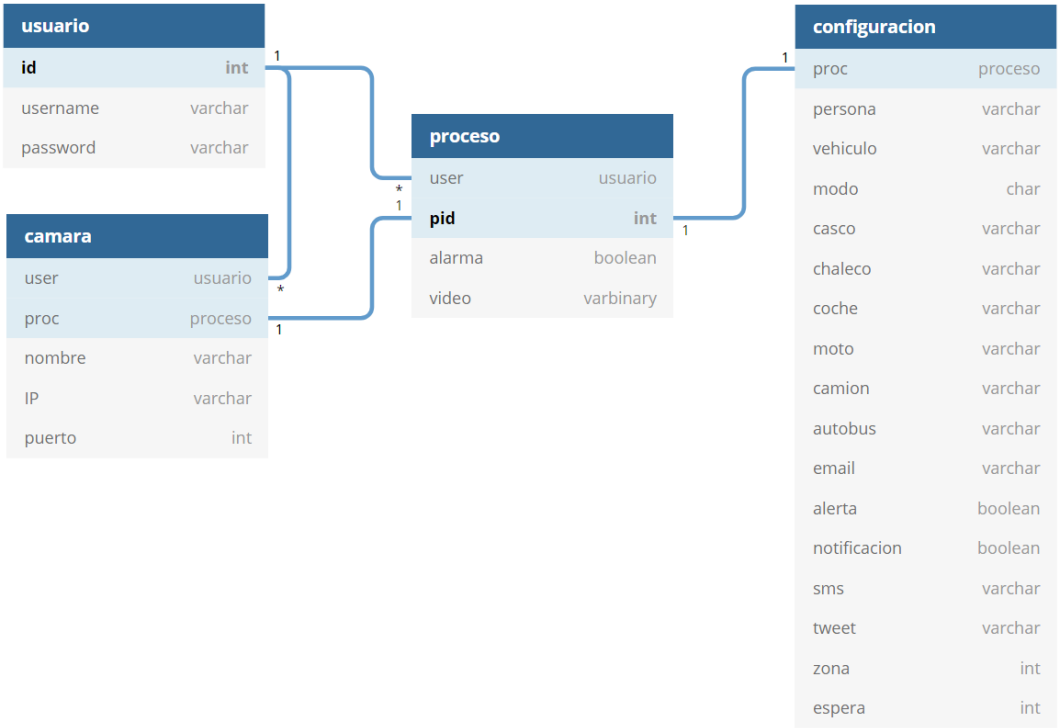

Figura 4.16: Diagrama de la base de datos con tablas y relaciones

La tabla usuario sigue el modelo de User de Django [16] pero ha sido simplificado en la figura para su mejor comprensión. Cada usuario de la aplicación está guardado en esta tabla con los datos fijados en su registro por el administrador. Esta tabla está relacionada con las tablas de cámara y proceso.

La relación uno a muchos con la tabla cámara refleja la posibilidad del usuario de añadir tantas cámaras como se desee a la aplicación para su posterior configuración y visualización. Mientras que, la relación uno a muchos de usuario con proceso muestra la posibilidad de que un usuario tenga varias cámaras activas en un momento dado [Figura 4.8]. La tabla cámara guarda en cada registro la información de retransmisión de las cámaras IP añadidas por el usuario, también guarda el nombre asignado como identificador. La cámara, además de estar relacionado con un usuario, está relacionada con un proceso si la cámara está activa, en caso contrario tomará el valor null. Esta relación permite

identificar a un proceso con la cámara que está siendo usada en él y viceversa, ya que una cámara no puede tener asignado más de un proceso y un proceso no puede estar vigilando más de una cámara.

En el lado opuesto de la relación, la tabla proceso guarda el *pid* del proceso que, además, es clave primaria de la tabla. Esta tabla es protagonista en la comunicación entre ambas aplicaciones de la que hablábamos al principio de la sección. El motivo de su importancia son los campos **alarma**, en el que se señala mediante un *boolean* si la alarma se activa, y **vídeo** en el que se van guardando todas las imágenes o *frames* que se analizan en la aplicación de vídeo y que son los mostrados en el apartado de visualización de vídeo de la aplicación web.

Por último, en la tabla de configuración se guardan las configuraciones realizadas por el usuario, estas configuraciones servirán en un futuro para poder reanudar una vigilancia que ha sido finalizada por el usuario y que en caso de querer repetir tendría que configurar desde el principio. Además, sus datos también son usados para mostrar al usuario en la pantalla de visualización los detalles de la configuración realizada [Figura 4.10].

5

Integración, pruebas y resultados

En este capítulo se presentan todas las pruebas realizadas a la aplicación tanto de rendimiento como de cumplimiento de los requisitos establecidos, en especial, en cuestión de detección de objetos y color.

5.1. Conexión y visualización

Las pruebas realizadas en el proyecto empezaron por la visualización de las distintas cámaras que pueden ser, ahora o en un futuro, usadas en la aplicación. Se buscaba, por tanto, conectar todos los distintos tipos de cámara y comprobar su correcto funcionamiento.

Se comprobó el funcionamiento de la webcam integrada del ordenador portátil donde se realizaba el desarrollo, una webcam externa conectada mediante un cable USB y una cámara IP que, en este caso, sería la cámara de un teléfono móvil accesible por IP mediante el uso de la aplicación DroidCam [17]. Todas ellas funcionaron correctamente.

5.2. Precisión en detección de objetos y color

La precisión con la que se detectan los objetos sobre la imagen es excepcional, reconociendo personas que solo cuentan con partes pequeñas del cuerpo dentro de la imagen en la gran mayoría de los casos. Esto es así dado que la librería de detección devuelve un porcentaje de seguridad sobre el objeto que ha reconocido siendo este alto si lo reconoce claramente y bajo si el objeto está parcialmente tapado o es difícil de reconocer por las condiciones de la imagen.

Además, se han realizado pruebas para hallar la distancia máxima de reconocimiento de la librería. Estas pruebas se realizaron sobre unas obras en las que se consiguió reconocer a una persona a una distancia de aproximadamente 40 metros pero no se reconoció a una persona a 50 metros. Esta prueba depende también de la calidad de la imagen que en este caso era una grabación en 720p, es decir, 1280 x 720 píxeles.

Figura 5.1: Detección de personas a distancias lejanas. Existen personas a la izquierda de la pareja de personas detectadas que no son marcadas debido a su lejanía.

Por otro lado, se han realizado pruebas de precisión sobre la detección de color en cascos y chalecos, para ello se han usado vídeos con personas que llevan cascos y chalecos de distintos colores. Esta detección es precisa en la mayoría de los casos, pero los fondos y colores comunes en imágenes como blancos o negros son problemáticos en la detección de color.

Cuando, por ejemplo, se desea detectar un casco marrón sobre un fondo en el que hay tierra la detección resulta en falsos positivos, esto también puede pasar con el color del cabello de la persona. Esta es una de las mayores áreas a mejorar, como veremos en el siguiente capítulo.

Sin embargo, cuando los colores son llamativos como suele ser habitual en los chalecos de seguridad la detección obtiene muy buenos resultados, acertando prácticamente en el 100 % de los casos.

5.3. Rendimiento

Las pruebas de rendimiento realizadas han sido sobre un ordenador portátil con procesador I7-7700HQ, 8GB de memoria RAM, tarjeta gráfica Nvidia GeForce GTX 950m y sistema operativo Ubuntu 18.04. El cuello de botella, en este caso, para la aplicación se sitúa en la tarjeta gráfica ya que la detección de objetos realizada por TensorFlow es realizada en su versión sobre GPU [18].

Sobre esta máquina se ha conseguido que una cámara activa funcione con fluidez aunque el ren-

dimiento con el tiempo decae y aparece retardo entre la imagen capturada y la imagen visualizada. Por otro lado, al poner en funcionamiento dos cámaras al mismo tiempo los problemas crecen y el rendimiento no es aceptable.

El rendimiento obtenido no es una sorpresa ya que la tarjeta gráfica usada es de gama baja y alguna de las tarjetas gráficas para usuarios domésticos actuales como la Nvidia RTX 2080 tienen un rendimiento hasta un 800 % superior [19]. Si optamos por la gama profesional Quadro de Nvidia la diferencia de rendimiento podría crecer hasta el 1100 % [20] con una sola tarjeta gráfica. Por lo que en un servidor preparado para la carga de procesamiento que las tareas realizadas suponen, las cámaras activas soportadas y el rendimiento de éstas crecería significativamente.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Al empezar el proyecto no estaba muy claro hasta dónde se podría llegar con tan pocas horas, como son las disponibles para la realización de un Trabajo de Fin de Grado, para desarrollar la aplicación. Empezando el proyecto de cero, el objetivo era obtener una aplicación básica, pero efectiva y usable, de detección de objetos que permitiera seguir el desarrollo de ésta en un futuro, incluso en otro Trabajo de Fin de Grado en los próximos años.

En las primeras fases del proyecto, se definieron los requisitos y diseño de la aplicación completa, aunque el tiempo pudiese no ser suficiente para completarlo. También se realizó la investigación de las tecnologías a utilizar en estas primeras fases.

La única fase de trabajo en común con compañeros de la empresa fue el análisis inicial en el que se definieron los requisitos y objetivos de la aplicación [apartados 1.2 y 3.1], además de algunas pruebas de validación realizadas por compañeros para evitar el sesgo propio. El diseño, investigación y desarrollo de la aplicación fue realizado, en exclusiva, por el autor del trabajo.

Durante el proyecto se ha fijado como principio documentar tanto el código como el funcionamiento de la aplicación para que el mantenimiento de la aplicación y el posible trabajo futuro sobre ella sea lo más sencillo posible. Por otro lado, se han usado tecnologías y procedimientos que se usaron en las prácticas realizadas en la empresa, facilitando así la integración de la aplicación en el ecosistema existente.

El resultado final ha cumplido con las expectativas y los requisitos se han podido cumplir con una alta satisfacción en general. En especial, los objetivos prioritarios de detección de objetos y definición de zonas. Sin embargo, todavía queda trabajo y mejoras por realizar, de las que hablaremos en la próxima sección.

6.2. Trabajo futuro

Durante el tiempo disponible para la realización del Trabajo de Fin de Grado se ha conseguido realizar una aplicación completa pero con margen de mejora, especialmente en dos terrenos, el rendimiento y la detección de color. La mejora en el primero depende, en parte, de la mejora del segundo.

La detección del color, como se comentó en el capítulo del desarrollo, da como resultado en ocasiones falsos positivos, cuando el color elegido puede ser encontrado en el fondo de la imagen. La solución a este problema pasa por eliminar el fondo de la imagen del objeto detectado, aunque también es posible ajustar la zona de detección al elemento buscado, sea casco o chaleco, para también evitar así otras zonas del cuerpo, pero esta solución puede ser más costosa computacionalmente.

Por otro lado, aprovechando la inclusión de las configuraciones en la base de datos, una de las futuras mejoras será la posibilidad de activar y parar configuraciones realizadas o cargar estas configuraciones desde la base de datos, evitando así la necesidad de repetir la configuración en futuras ocasiones.

Para mejorar el rendimiento, además de mejorar la detección de color, se mejorará el cálculo de la intersección entre zona y objeto, intentando reducir los cálculos realizados al mínimo imprescindible, ya que se realiza una intersección por cada objeto detectado en cada frame, con lo que cualquier mejora, por pequeña que sea, tendrá un impacto importante en el rendimiento. También se dejará de usar la librería ImageAI para pasar a utilizar las librerías que están por debajo como TensorFlow o Keras, esto permitirá mayor control sobre las tareas realizadas. Además, se estudiará la opción de migrar la aplicación a C++ y abandonar Python, lo que podría mejorar el rendimiento global.

En resumen, el trabajo futuro es amplio ya que, aunque se ha logrado obtener una aplicación con todas las características que se deseaban en su concepción, el tiempo disponible para llevarlas a cabo ha sido limitado y por tanto, estas características todavía pueden ser ampliadas y perfeccionadas.

Bibliografía

[1] https://www.nchsoftware.com/surveillance/es/index.html.

[2] https://teboweb.com/api/TeboCam/Index.

[3] http://www.ugolog.com.

[4] https://www.sighthound.com.

[5] https://www.sighthound.com/products/sighthound-video.

[6] https://www.sighthound.com/technology/.

[7] https://www.netatmo.com/es-es/security/cam-outdoor.

[8] https://www.netatmo.com/es-es.

[9] https://shop.netatmo.com/eur_es/presence.html.

[10] M. Yang and K. Yu., "Real-time clothing recognition in surveillance videos," *ICIP*, 2011.

[11] https://developer.mozilla.org/es/docs/Web/API/Notifications_API.

[12] https://github.com/OlafenwaMoses/ImageAI.

[13] https://docs.djangoproject.com/en/2.2/topics/auth/default/.

[14] https://docs.python.org/3/library/argparse.html.

[15] https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html.

[16] https://docs.djangoproject.com/en/2.2/ref/contrib/auth/#user-model.

[17] https://play.google.com/store/apps/details?id=com.dev47apps.droidcam.

[18] https://www.tensorflow.org/install/gpu.

[19] https://gpu.userbenchmark.com/Compare/Nvidia-RTX-2080-vs-Nvidia-GTX-950M/4026vsm27713.

[20] https://gpu.userbenchmark.com/Compare/Nvidia-Quadro-RTX-6000-vs-Nvidia-GTX-950M/m736712vsm27713.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID